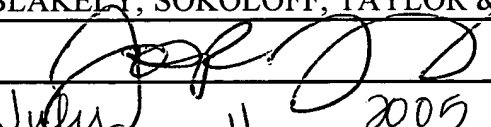
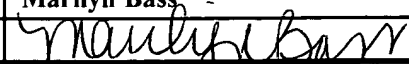


AF IFW

TRANSMITTAL FORM (to be used for all correspondence after initial filing)		Application No.	09/505,949
		Filing Date	February 15, 2000
		First Named Inventor	Michael Chow
		Art Unit	2183
		Examiner Name	Li, Aimee J.
Total Number of Pages in This Submission	30	Attorney Docket Number	42390P6447

ENCLOSURES (check all that apply)		
<input checked="" type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Response <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> PTO/SB/08 <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/Incomplete Application <input type="checkbox"/> Basic Filing Fee <input type="checkbox"/> Declaration/POA <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Drawing(s) <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s)	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input checked="" type="checkbox"/> Appeal Communication to Group (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input checked="" type="checkbox"/> Other Enclosure(s) (please identify below): <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">Return receipt postcard</div>
Remarks Response to Notification of Non-Compliant Supplemental Appeal Brief Under 37 CFR 41.37, w/ Exhibit 1		

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Joseph Lutz, Reg. No. 43,765 BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Signature	
Date	July 11, 2005

CERTIFICATE OF MAILING/TRANSMISSION			
I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.			
Typed or printed name	Marilyn Bass		
Signature		Date	07-11-05



FEE TRANSMITTAL for FY 2005

Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27.

Complete if Known

Application Number	09/505,949
Filing Date	February 15, 2000
First Named Inventor	Michael Chow
Examiner Name	Li, Aimee J.
Art Unit	2183
Attorney Docket No.	42390P6447

TOTAL AMOUNT OF PAYMENT (\$)

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit card ☐ Money Order ☒ None ☐ Other (please identify): _____

☒ Deposit Account Deposit Account Number: 02-2666 Deposit Account Name: Blakely, Sokoloff, Taylor & Zafman LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☐ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, except for the filing fee
☒ Charge any additional fee(s) or underpayment of fee(s) under 37 CFR §§ 1.16, 1.17, 1.18 and 1.20. ☐ Credit any overpayments

FEE CALCULATION

Large Entity		Small Entity		Fee Description	Fee Paid
Fee Code	Fee (\$)	Fee Code	Fee (\$)		
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
2053	130	2053	130	Non-English specification	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	
1403	1,000	2403	500	Request for oral hearing	
1451		2451		Petition to institute a public use proceeding	
1460	130	2460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
1809	790	1809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
Other fee (specify) _____					
SUBTOTAL (2)					(\$)

SUBMITTED BY

Complete (if applicable)

Name (Print/Type)	Joseph Lutz	Registration No. (Attorney/Agent)	43,765	Telephone	(310) 207-3800
Signature		Date	07-11-05		



Attorney's Docket No.: 042390.P6447

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application for:

Michael Chow, et al.

Application No.: 09/505,949

Filed: February 15, 2000

For: **METHOD AND APPARATUS FOR
ACHIEVING ARCHITECTURAL
CORRECTNESS IN A MULTI-MODE
PROCESSOR PROVIDING FLOATING-
POINT SUPPORT**

Examiner: Li, Aimee J.

Art Group: 2183

**RESPONSE TO NOTIFICATION OF NON-COMPLIANT
SUPPLEMENTAL APPEAL BRIEF UNDER 37 C.F.R. §41.37**

Mail Stop Appeal Brief - Patent
Commissioner for Patents
P. O. 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Notification of Non-Compliant Appeal Brief mailed on June 16, 2005, Applicants submit the following Supplemental Appeal Brief pursuant to 37 C.F.R. §41.37 for consideration by the Board of Patent Appeals and Interferences. Should any charges be required, please charge any additional amount due or credit any overpayment to deposit Account No. 02-2666.



TABLE OF CONTENTS

	Page
I. REAL PARTY IN INTEREST	2
II. RELATED APPEALS AND INTERFERENCES.....	2
III. STATUS OF CLAIMS.....	2
IV. STATUS OF AMENDMENTS.....	2
V. SUMMARY OF THE CLAIMED SUBJECT MATTER.....	2
VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL.....	4
VII. ARGUMENT	4
A. <u>Overview of the Cited References</u>	4
1. <u>Overview of Blomgren Reference</u>	4
2. <u>Overview of Blomgren2 Reference</u>	5
3. <u>Overview of FOLDLOC Reference</u>	6
4. <u>Overview of Omondi Reference</u>	7
B. <u>Rejection of Claims 1, 2, 4, 5, 7, 8 and 18 as Anticipated by Blomgren and Blomgren2</u>	7
1. <u>Errors of Law and Fact in the Rejection</u>	7
2. <u>Specific Limitations Not Described in the Prior Art</u>	9
3. <u>Explanation Why Such Limitations Render the Claims Non-obvious over the Prior Art</u>	9
C. <u>Rejection of Claim 3 As Anticipated by Blomgren and Blomgren2</u>	10
1. <u>Errors of Law and Fact in the Rejection</u>	10
2. <u>Specific Limitations Not Described in the Prior Art</u>	11
3. <u>Explanation Why Such Limitations Render the Claims Unanticipated by the Prior Art</u>	11
D. <u>Rejection of Claims 9 and 17 As Anticipated by Blomgren and Blomgren2</u>	12
1. <u>Errors of Law and Fact in the Rejection</u>	12
2. <u>Specific Limitations Not Described in the Prior Art</u>	14
3. <u>Explanation Why Such Limitations Render the Claims Unanticipated by the Prior Art</u>	14
E. <u>Rejection of Claims 10-13, 15, 16 and 19 As Anticipated by Blomgren</u>	14
1. <u>Errors of Law and Fact in the Rejection</u>	14
2. <u>Specific Limitations Not Described in the Prior Art</u>	15
3. <u>Explanation Why Such Limitations Render the Claims Unanticipated by the Prior Art</u>	15

	Page
F. <u>Rejection of Claim 6 and 14 As Anticipated by Blomgren and Blomgren2</u> <u>in View of FOLDOC</u>	16
1. <u>Errors of Law and Fact in the Rejection</u>	16
2. <u>Specific Limitations Not Described in the Prior Art</u>	18
3. <u>Explanation Why Such Limitations Render the Claims Non-obvious</u> <u>over the Prior Art</u>	18
G. <u>Rejection of Claim 18 As Obvious Over Blomgren and Blomgren2 in</u> <u>View of Omondi</u>	19
1. <u>Errors of Law and Fact in the Rejection</u>	19
2. <u>Specific Limitations Not Described in the Prior Art</u>	20
3. <u>Explanation Why Such Limitations Render the Claims Non-obvious</u> <u>over the Prior Art</u>	20
VIII. CLAIMS APPENDIX	22



I. REAL PARTY IN INTEREST

Michael Chow, Elango Ganesan, John William Phillips and Nazar Abbas Zaidi, the parties named in the caption, transferred their rights to that which is disclosed in the subject application through an assignment recorded on March 20, 2000 (010760/0706) in the patent application to Intel Corporation, of Santa Clara, California. Thus, as the owner at the time the brief is being filed, Intel Corporation, of Santa Clara, California is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences which will affect or be affected by the outcome of this appeal.

III. STATUS OF CLAIMS

Claims 1-19 are pending and rejected in this application. Applicants hereby appeal the rejection of all pending claims.

IV. STATUS OF AMENDMENTS

The claims are amended in accordance with the Response Amendment filed on August 8, 2003, wherein Claim 18 was amended. The claim amendments requested in the Response Amendment filed on January 15, 2004 regarding Claims 10 and 19 were not entered.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Claims 1 and 18 recite a multi-mode processor 100 to process instructions from a first instruction set architecture (ISA) 110 having a first word size and to process instructions from a second ISA 120 having a second word size, where the second word size of the second ISA is different than the first word size of the first ISA, as shown in FIG. 1. As described at page 7 of Applicants' specification,

In one embodiment, a processor is capable of operating in two modes. The first and second modes are a 32 bit word ISA and a 64 bit word ISA, respectively. More specifically, the first mode is IA-32 mode, in which the processor emulates a 32 bit word Intel architecture (IA). . . . [T]he second mode is IA-64, which implements what is known as the IA-64 ISA. (pg. 7, lines 1-16.)

As recited by Claim 9, in one embodiment, the multi-mode processor 100 provides backward compatibility or legacy support for a 32 bit word (legacy) ISA, as well as support for a 64-bit word (current) ISA. As described, the term "word size", as known to those skilled in the art, refers to the number of bits that a CPU can process at one time. In one embodiment, the processor includes floating point registers 150 and floating point units 160, which are shared between a legacy ISA engine 110/120 and a current ISA engine 120/110, as illustrated with

reference to FIG. 1. In a computer's arithmetic logic unit (ALU), there sometimes exists input encodings, which the machine must interpret as tokens, which require special handling. To provide support for such encodings, the current ISA includes a special FP encoding ("NaTVal"), as recited by Claim 6. When such a value is detected, the processor known value causes a floating point unit to ignore the requested operation and propagate the NaTVal as output, typically causing the processor to later request the data and/or operation non-speculatively when it is needed. (See pg. 8, lines 2-19.)

Unfortunately, the legacy ISA does not support NaTVal tokens. However, because the floating point registers and floating point units are shared between the legacy ISA engine and current ISA engine of the multi-mode processor, as recited by Claim 3, in one embodiment, as illustrated with reference to FIG. 2, preprocessing hardware 162 detects whether a NaTVal token (or other token and special values) is present in input operands. When a NaTVal token or other special token is detected by preprocessing hardware 166, depending on what mode the processor is in, values other than a true arithmetic result are prepared by post-processing hardware 166. (See pg. 11, lines 4-17.)

Operation of the multi-mode processor is illustrated with reference to FIG. 3. As recited by Claims 10 and 19, at process block 320, preprocessing hardware analyzes input operands to detect which input operands must be interpreted as tokens (NaTVal) instead of being fed to the arithmetic unit. Also, where a result is produced, at process block 350, post-processing hardware replaces the arithmetic result with the expected result given the special input tokens. However, as indicated at process block 340, the capability to inject the special result is turned on or off depending on whether the multi-mode processor is processing instructions from the legacy ISA or the current ISA which supports NaTVal tokens. Representatively, if the processor is in a first mode, the requested calculation is performed at process block 330. Otherwise, an NaTVal specific operation is performed at process block 340.

As recited by Claim 14, operation of the multi-mode process includes detecting whether a NaTVal token (or other special token and special values) is present on input operand. When a NaTVal token or other special token is detected by, for example, preprocessing hardware 166, depending on what mode the processor is in, values other than the true arithmetic result are prepared by, for example, post-processing hardware 166. (See, pg. 11, lines 4-17 and FIG. 2.)

As recited by Claim 17, operation of the multi-mode processor supports a legacy ISA having, for example, a 32 bit word size and a current ISA, having a 64 bit word size.

VI. GROUND OF REJECTION TO BE REVIEWED ON APPEAL

Are Claims 1-5, 7-13 and 15-19 unpatentable under 35 U.S.C. §102(b) as being taught by Blomgren et al., U.S. Patent No. 5,685,009 ("Blomgren") and U.S. Patent No. 5,781,750 ("Blomgren2")?

Are Claims 6 and 14 unpatentable under 35 U.S.C. §103(a) as being obvious over Blomgren and Blomgren2 in view of Instant Web's Online Computing Dictionary terms "speculative evaluation" and "speculative execution" ("FOLDOC")?

Is Claim 18 unpatentable under 35 U.S.C. §103(a) as being obvious over Blomgren and Blomgren2 in view of Amos Omondi's *The Microarchitecture of Pipelined and Superscalar Computers* ©1999 ("Omondi")?

VII. ARGUMENT

A. Overview of the Cited References

1. Overview of Blomgren Reference

Blomgren teaches a mechanism for sharing registers between a RISC architecture and a CISC on a dual instruction set CPU. (Col. 2, lines 22-23.) As indicated in Blomgren:

Two instruction decoders are required when the instruction sets are separate because the instruction sets each have an independent encoding of operations to opcodes. For example, both instruction sets have an ADD operation or instruction. However, the binary opcode number which encodes the ADD operation is different for the two instruction sets. In fact, the size and location of the opcode field in the instruction word is also different for the two instruction sets. In the x86 CISC instruction set, the opcode 03 hex is the ADD r,v operation or instruction for a long operand. This same opcode, 03 hex, corresponds to a completely different instruction in the PowerPC™ RISC instruction set. (Col. 1, lines 44-56.)

To this end, Blomgren describes various mechanisms for sharing registers between a CISC architecture and a RISC architecture to provide a shared register architecture, as depicted with reference to FIG. 8.

As illustrated with reference to FIG. 8:

The CISC EFLAGS register and the RISC CR register are combined into a single 32-bit CP/EFLAGS register 40 that can be accessed by CISC user programs and RISC user programs and emulation code. The CISC code segment base address register and the RISC count CTR register are merged to a single CS/CTR register 42, also accessible by CISC user programs and RISC user programs and emulation code. The RISC system save/restore (SRR0) register, which normally holds the address to return to after an interrupt has been processed, also holds the return address when emulation code was called. ...

The RISC link register, ... is combined with a CISC register that holds the instruction address of the most recent floating point instruction. ... RISC and emulation programs can freely access this merged FP-IP/LR register 46. The 32 general-purpose registers from RISC are merged with the 8 GPR's and 6 segment base registers from the CISC architecture into merged GPR's 48. ...

Data registers for numbers in floating-point format are also merged into one set of floating-point data register file 60 for both RISC and CISC programs. Status and control bits for floating point operations in both RISC and CISC modes are combined into a single floating point status and control register FPSCR register 52. (Col. 19, line 48 - Col. 20, line 15.)

The shared register architecture shown in FIG. 8 enable Blomgren to provide emulation mode for execution of complex CISC instructions:

... emulation mode, also uses the first instruction decoder for RISC instructions, but emulation mode executes a superset of the RISC instruction set. Using emulation mode, individual CISC instructions may be emulated with RISC instructions. Thus, not all CISC instructions need to be directly supported in the CPU's hardware. Unsupported CISC instructions cause a jump to an emulation mode routine to emulate the unsupported CISC instruction. Upon completion of the emulation mode routine, control is returned to the CISC program with the next CISC instruction. (Col. 3, lines 57-67.) (Emphasis added.)

Accordingly, as taught by Blomgren:

In the dual-instruction-set processor, when a CISC program causes an exception, a RISC emulation routine is called and executed. Thus the normal CISC exception handling hardware is not needed and no counterpart in the SRR0 register is necessary. (Col. 14, lines 1-5.)

For example, as taught by Blomgren:

This complex CISC AAA instruction is not supported by the instruction decoder and signals an unsupported opcode exception, which causes emulation mode to be entered from CISC mode. An emulation routine of RISC instructions is executed to emulate the complex CISC instruction. (col. 8, line 66 - col. 9, line 4.)

In other words, Blomgren teaches the issuance of an unsupported opcode exception in response to detection of an unsupported CISC instruction and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation routine including various RISC instructions to perform the CISC instruction. Once the exception handling is complete, the mode is switched to the CISC mode and a next CISC instruction is executed. (See, col. 14, lines 22-28.)

2. Overview of Blomgren2 Reference

Blomgren2 is incorporated by reference within Blomgren. (See, Blomgren, col. 3, lines 44-47.) Blomgren2 teaches a dual instruction set architecture CPU with hidden software emulation mode. (See, Abstract and Fig. 2.) As indicated in Blomgren2:

A comparison of the opcode decoding of Table 2 for the RISC instruction set with Table 3 for the CISC instruction set shows that the two sets have independent encoding of instructions to opcodes. While both sets have ADD operations, the opcode number which encodes the ADD operation is different for the two instruction sets. In fact, the size and location of the opcode field in the instruction word is also different for the two instruction sets. (Col. 9, lines 16-20.) (Emphasis added.)

As a result:

Two separate decode blocks are necessary for the two separate instruction sets. (Col. 9, lines 13-14.)

To this end, Blomgren2 teaches an emulation mode:

Emulation mode runs a super-set of the RISC instruction set. Additional “extended” instructions are added for controlling the CPU’s hardware, such as direct access to the TLB, register files, cache, and selecting between the three operating modes of the CPU. Emulation mode executes routines that emulate the behavior of the complex instructions that are not supported directly by the hardware. (Col. 5, lines 49-55.) (Emphasis added.)

As illustrated in FIG. 2 of Blomgren2:

Mode control logic 42 causes emulation mode to be entered whenever a miss is signaled from TLB 52, or an unknown opcode is detected by instruction decode unit 36. Normal exceptions, interrupts, and traps from the execute unit and other units also cause emulation mode to be entered, giving great flexibility in system design. Mode control logic 42 sets and clears the RISC/CISC and emulation mode control bits in mode register 38. When entry to emulation mode is requested, entry point block 56 generates the proper entry point vector or address in the emulation portion of memory, and loads this address into the instruction pointer 34. Thus the CPU will begin fetching and executing instructions at the specified entry point, where the emulation driver contains a routine to handle the exception, TLB miss, or to emulate the unknown instruction. (Col. 7, lines 30-43.) (Emphasis added.)

In other words, Blomgren2 teaches the detection of an unknown opcode and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation driver to perform the CISC instruction. Once completed, the emulation code returns control to the CISC mode, which causes a user program to continue execution with a next instruction. (See, col. 5, lines 57-65.)

3. Overview of FOLDOC Reference

FOLDOC describes definitions of speculative evaluation and speculative execution.

As described by FOLDOC, speculative evaluation is defined as:

A technique used in parallel processing where some evaluation may be started before it is known whether it is needed (eager evaluation). This may result in some wasted processing and may introduce unnecessary non-terminating processes, but it can reduce the overall run time by making some needed results available earlier than they would be otherwise.

FOLDOC defines speculative execution as:

A technique allows a superscalar processor to keep its functional units as busy as possible by executing instructions before it is known that they will be needed.

4. Overview of Omondi Reference

Omondi describes the fundamentals of pipelining. As described by Omondi:

The basic technique used to obtain high performance in the design of pipeline machines is the same one used to obtain high productivity in factory assembly lines. In the latter situation, the work involved in the production of some object is partitioned among a group of workers arranged in a linear order, such that each worker performs a particular task in the production process before passing the partially completed product down to the next worker in the line. All workers operate concurrently, and a completed product is available at the end of the line. This arrangement is an example of temporal parallelism - that is parallelism in time - and appears in several different forms and in the designs of high performance computers. The two basic forms of pipelining are instruction pipelining and arithmetic pipelining. (See, pp. 1 and 2, ¶1 of Omondi.)

In addition to describing the fundamentals of pipelining, Omondi illustrates a DEC Alpha 21164 processor pipeline in FIG. 1.9, a DEC Alpha 21264, processor pipeline in FIG. 1.11, a PowerPC 604 processor pipeline in FIG. 1.12 and a power3 processor pipeline in FIG. 1.13. As shown in FIG. 1.11, the DEC Alpha 21264 processor pipeline includes two floating point arithmetic logic units, whereas the PowerPC 604 processor pipeline, as shown in FIG. 1.12, includes a single floating point unit and the power3 processor pipeline, as shown in FIG. 1.13 includes two floating point units. Also, the DEC Alpha 21164 processor pipeline, as shown in FIG. 1.9, includes two floating point pipelines.

B. Rejection of Claims 1, 2, 4, 5, 7, 8 and 18 as Anticipated by Blomgren and Blomgren2

The Examiner rejected all pending claims, including Claims 1, 2, 4, 5, 7, 8 and 18 under 35 U.S.C. §102 (b) as anticipated by Blomgren, which incorporates Blomgren2 by reference.

1. Errors of Law and Fact in the Rejection

Applicant respectfully asserts that the Examiner has failed to adequately set forth a *prima facie* rejection under 35 U.S.C. §102(b). “Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, *arranged as in the claim.*” Lindemann Maschinenfabrik v. American Hoist & Derrick (“Lindemann”), 730 F.2d 452, 1458 (Fed. Cir. 1994)(emphasis added). Additionally, each and every element of the claim must be exactly disclosed in the anticipatory reference. Titanium Metals Corp. of American v. Banner (“Banner Titanium”), 778 F.2d 775, 777 (Fed. Cir. 1985).

Although the Examiner has rejected Claims 1, 2, 4, 5, 7, 8 and 18 as anticipated by Blomgren and Blomgren2, the Examiner fails to show that either Blomgren or Blomgren2 teach the claimed subject matter.

According to the Examiner, Blomgren (col. 1, lines 24-33 and 44-48 and col. 1, lines 59 to col. 2, line 4) and Blomgren2 (Abstract; col. 3, lines 51-55 and 59-65; col. 6, lines 16-24; and FIG. 2) teach first and second instructions set engines to process instructions having a first word size and a second word size, the second word size being different than the first word size. (See pg. 3 of the Office Action mailed 12/01/04.) However, after having carefully reviewed the relevant portions of Blomgren and Blomgren2 cited by the Examiner, Applicants must respectfully disagree with the Examiner's contention.

Applicants submit that the Examiner is improperly equating the term "instruction size" with the term "word size", as recited by independent Claims 1 and 18. As defined by the Computer Dictionary Online, the term "word size" refers to the number of bits that a CPU can process at one time. (See Exhibit 1) Applicants submit that the number of bits a CPU can process at one time does not vary depending on whether the CPU is processing CISC instructions or RISC instructions.

Furthermore, Applicants submit that there is no suggestion as to any variation in the word sizes between the RISC and CISC instruction sets within either Blomgren, Blomgren2 or the references of record.

Specifically, as described in Blomgren:

Two instruction decoders are required when the instruction sets are separate because the instruction sets each have an independent encoding of operations to opcodes. For example, both instruction sets have an ADD operation or instruction. However, the binary opcode number which encodes the ADD operation is different for the two instruction sets. In fact, the size and location of the opcode field in the instruction word is also different for the two instruction sets. In the x86 CISC instruction set, the opcode 03 hex is the ADD r,v operation or instruction for a long operand. This same opcode, 03 hex, corresponds to a completely different instruction in the PowerPC™ RISC instruction set. (Col. 1, lines 44-56.)

Likewise, as described in Blomgren2:

A comparison of the opcode decoding of Table 2 for the RISC instruction set with Table 3 for the CISC instruction set shows that the two sets have independent encoding of instructions to opcodes. While both sets have ADD operations, the opcode number which encodes the ADD operation is different for the two instruction sets. In fact, the size and location of the opcode field in the instruction word is also different for the two instruction sets. (Col. 9, lines 16-20.) (Emphasis added.)

Applicants submit that the cited passages imply that the instruction word size is the same for both the CISC and RISC instruction sets, and in accordance with the definition of "word size" provide above. Unless the CISC and RISC instruction words are the same size, indication of "the size and location of the opcode field in the instruction word being different for the two instruction sets," as explicitly stated in both Blomgren and Blomgren2, makes no sense. In other words, use of the definite article in this context requires a single instruction word size. Accordingly,

one skilled in the art would not interpret either Blomgren or Blomgren2 as teaching instruction set engines for processing instructions from ISAs having different word sizes.

Applicants submit that the entire description of both Blomgren and Blomgren2 is devoid of any reference to providing processing for different ISAs having different ISA word sizes. The case law is quite clear in establishing that each and every element of the claim must be exactly disclosed in the anticipatory reference. Banner Titanium, Id. Hence, a *prima facie* case of anticipation of the claims over Blomgren and Blomgren2 has not been established and the rejection of Claims 1, 2, 4, 5, 7, 8 and 18 is therefore erroneous.

2. Specific Limitations Not Described in the Prior Art

Each of independent Claims 1 and 18 recite:

- (1) a first instruction set engine to process instructions from a first ISA having a first word size;
- (2) a second instruction set engine to process instructions from a second ISA having a second word size, the first word size being different than the second word size. (Emphasis added.)

3. Explanation Why Such Limitations Render the Claims Non-obvious over the Prior Art

Applicants claim a multi-mode processor that includes a first instruction set engine to process instructions from a first instruction set architecture ISA having a first word size. The multi-mode processor also includes a second instruction set engine to process instructions from a second ISA having a second word size, the second word size being different than the first word size, as recited by independent Claims 1 and 18. As described at pg. 7 of Applicants' specification:

In one embodiment, a processor is capable of operating in two modes. The first and second modes are a 32 bit word ISA and a 64 bit word ISA, respectively. More specifically, the first mode is IA-32 mode, in which a processor emulates a 32 bit word Intel Architecture (IA) known as IA-32 ISA . . . [T]he second mode is a IA-64, which implements what is known as the IA-64 ISA. (pg. 7, lines 1-16.) (Emphasis added.)

Accordingly, in one embodiment, the multi-mode processor provides backward compatibility or legacy support for a 32 bit word (legacy ISA), as well as support for a 64 bit word (current) ISA.

In contrast, Blomgren teaches the issuance of an unsupported opcode exception in response to detection of an unsupported CISC instruction and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation routine including various RISC instructions to perform the CISC instruction. Once the exception handling is complete, the mode is switched to the CISC mode and a next CISC instruction is executed. (See, col. 14, lines 22-28.)

Likewise, Blomgren2 teaches the detection of an unknown opcode and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation driver to perform the CISC instruction. Once completed, the emulation code returns control to the CISC mode, which causes a user program to continue execution with a next instruction. (See, col. 5, lines 57-65.)

Hence, Applicants submit that the entire specification of Blomgren and Blomgren2 are devoid of any reference to providing processing for different ISAs having different ISA word sizes. Applicants submit that the explicit text of Blomgren (See, col. 1, lines 44-56) and Blomgren2 (See, col. 9, lines 16-20) imply that the instruction word size is the same for both the CISC and RISC instruction sets, and in accordance with the definition of “word size” provide above.

Yet, in spite of the lack of any teaching toward processing of ISAs having a different word size, by improperly equating the term “instruction size” with the term “word size”, the Examiner incorrectly finds a teaching within Blomgren and Blomgren2 to anticipate independent Claims 1 and 18. However, the case law is quite clear in establishing that each and every element of the claim must be exactly disclosed in the anticipatory reference. Banner Titanium, Id.

Therefore, a *prima facie* case of anticipation of the claims is not established and the rejection of Claims 1, 2, 4, 5, 7, 8 and 18 is erroneous and should be overturned. Accordingly, Applicants respectfully request that the §102(b) rejection of the Claims 1, 2, 4, 5, 7, 8 and 18 be overturned.

C. Rejection of Claim 3 As Anticipated by Blomgren and Blomgren2

The Examiner rejected all pending claims, including Claim 3 under 35 U.S.C. §102(b) as being anticipated by Blomgren and Blomgren2.

1. Errors of Law and Fact in the Rejection

The Examiner has made the same errors as previously described with respect to the rejected Claims 1, 2, 4, 7, 8 and 18. In addition, the Examiner has failed to show that each and every element of Claim 3 is exactly disclosed by Blomgren and Blomgren2. Banner Titanium, Id.

According to the Examiner, Blomgren and Blomgren2 teach pre-processing hardware to determine whether a token is received as an input and post-processing hardware to perform a token-specific operation, as recited by Claim 3. As indicated in the Office Action mailed December 14, 2004, Blomgren teaches the pre-processing hardware at col. 1, lines 43-47 and col. 2, lines 51-58, while Blomgren2 teaches pre-processing hardware at col. 3, line 65 to col. 4, line 21; col. 6, lines 53-59; col. 7, lines 30-54 and FIG. 2. After having carefully reviewed the cited passages, Applicants must respectfully disagree with the Examiner’s contention.

According to the Examiner, remote control logic 42, which causes emulation mode to be entered whenever an unknown opcode is detected by instruction decode unit 36, as taught by

Blomgren2, teaches the pre-processing hardware, as recited by Claim 3. (See, Blomgren2, col. 7, lines 30-43.) Applicants respectfully submit that the detection of an unknown opcode, as taught by Blomgren2, neither teaches nor suggests pre-processing hardware to detect whether a token is received as an input.

Furthermore, the passages cited by the Examiner to show pre-processing hardware within Blomgren refer to the sharing of data between registers (See, col. 2, lines 51-58 of Blomgren) and the use of two decode units for two ISAs that provide separate encodings (See, Blomgren, col. 1, lines 43-47). However, after carefully having reviewed these passages, Applicants respectfully submit that these passages clearly do not teach pre-processing hardware to detect whether a token is received as an input, as recited by Claim 3.

As further indicated by the Examiner, the emulation mode, as taught by Blomgren2 (See, col. 5, line 49 to col. 6, line 13), as well as the emulation as mode taught at col. 3, line 57-65 of Blomgren, teach post-processing hardware to perform a token-specific operation. Applicants respectfully submit that the emulation mode, as taught by Blomgren and Blomgren2, which teach the execution of routines that emulate the behavior of complex instructions that are not supported directly by hardware, does not teach or suggest post-processing hardware to perform a token-specific operation, as recited by Claim 3.

Applicants submit that the cited passage provides no reference to pre-processing hardware to determine whether a token is received as an input or post-processing hardware to perform a token specific operation, as recited by Claim 3. Hence, a *prima facie* case of anticipation of Claim 3 has not been established and the rejection of Claim 3 is therefore erroneous. *Id.*

2. Specific Limitations Not Described in the Prior Art

Claim 3 recites:

pre-processing hardware to detect if a token exists in the input;
an arithmetic unit responsive to the input and the mode identifier; and
post-processing hardware to perform a token specific operation if a
token exists in the input. (Emphasis added.)

3. Explanation Why Such Limitations Render the Claims Unanticipated by the Prior Art

The Applicants claim a processor including pre-processing and post-processing hardware to process an input to render an arithmetic result if a processor is in a first mode and performing a token specific operation if a processor is in a second mode when a token is detected as an input, as recited by Claim 3.

In contrast, Blomgren teaches the issuance of an unsupported opcode exception in response to detection of an unsupported CISC instruction and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation routine including various RISC instructions to perform the CISC instruction. Once the

exception handling is complete, the mode is switched to the CISC mode and a next CISC instruction is executed. (See, col. 14, lines 22-28.)

Likewise, Blomgren2 teaches the detection of an unknown opcode and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation driver to perform the CISC instruction. Once completed, the emulation code returns control to the CISC mode, which causes a user program to continue execution with a next instruction. (See, col. 5, lines 57-65.)

In other words, both Blomgren and Blomgren2 teach the detection of an unknown opcode and switch to an emulation mode when such an opcode is detected. However, the passages cited by the Examiner, as well as the entire text of Blomgren and Blomgren2 are devoid of any reference to detecting whether an input from a plurality of floating point registers includes a token, as recited by Claim 3.

Furthermore, Blomgren and Blomgren2 are devoid of any teachings for performing token specific processing by post-processing hardware if a token is received as an input. In contrast, Blomgren and Blomgren2 teach mode control logic that detects an unknown opcode which causes entry into emulation mode, which the Examiner equates to a token, to load pointers with the address of an emulation routine. (See, Blomgren2, col. 7, lines 30-43.)

Accordingly, although Blomgren and Blomgren2 describe an emulation mode to emulate the unknown instruction, the processing performed within Blomgren and Blomgren2 does not vary according to a processor mode as indicated by a mode identifier. Applicants respectfully submit that the detection of an unknown opcode and entry into emulation to emulate an unknown instruction does not teach the detection of whether an input contains a token or performing a token-specific operation according to a processor mode, as recited by Claim 3.

Therefore, a *prima facie* case of anticipation of the claims is not established and the rejection of Claim 3 should be overturned. Accordingly, Applicants respectfully request that the §102(b) rejection of Claim 3 be overturned.

D. Rejection of Claims 9 and 17 As Anticipated by Blomgren and Blomgren2

The Examiner rejected all pending claims, including Claims 9 and 17 under §102(b) as anticipated by Blomgren and Blomgren2.

1. Errors of Law and Fact in the Rejection

The Examiner has made the same errors as described previously with respect to the rejected Claims 1, 2, 4, 7, 8 and 18.

According to the Examiner, Blomgren and Blomgren2 have taught a mode identifier that indicates whether a processor is in a 32 bit word ISA mode or a 64 bit word ISA mode. (See,

pg. 6 of Office Action mailed 12/14/04.) To support the Examiner's argument, the Examiner states:

The powerPC instruction mode, i.e., RISC instruction mode, has 32 bit instruction words, as is the nature of the powerPC instruction set. The X instruction mode, i.e., CISC construction mode, has variable length instructions, which includes the 64 bit length instruction. For more information, please see the provided information for more information on the X86 and powerPC instruction sets in Hearing and Jordan's Computer System Design and Architecture provided with the Action dated 18 June 2003 about the RISC and CISC instruction sets. (See, pg. 6, ¶3 of Office Action mailed 12/14/04.)

After having reviewed the relevant passages of Blomgren and Blomgren2 cited by the Examiner, as well as the information on the X86, powerPC instruction set and Computer System Design and Architecture, Applicants respectfully submit that the Examiner is improperly equating the term "instruction size" with the term "word size," as recited by Claims 9 and 17.

As indicated by the passage above, the RISC instruction mode, as correctly indicated by the Examiner, includes 32 bit instruction words, as the word size of the processor is 32 bits. Furthermore, although the instruction length of the CISC instruction mode may include variable length instructions, which are 64 bits in length, the word size to process such an instruction, as seen by the CPU, remains at 32 bits. In other words, as defined by Exhibit 1, the term "word size" refers to the number of bits that a CPU can process at one time. Applicants submit that the number of bits a CPU can process at one time does not vary depending on whether the CPU is processing CISC instructions or RISC instructions.

Furthermore, as explicitly stated in both Blomgren (See, col. 1, lines 44-56) and Blomgren2 (See, col. 9, lines 16-20):

The size and location of the opcode field and the instruction word is also different for the two instruction sets.

Applicants submit that the cited passages imply that the instruction word size is the same for both the CISC and RISC instruction sets, and in accordance with the definition of "word size" provide above. Unless the CISC and RISC instruction words are the same size, indication of "the size and location of the opcode field in the instruction word being different for the two instruction sets," as explicitly stated in both Blomgren and Blomgren2, makes no sense. In other words, use of the definite article in this context requires a single instruction word size. Accordingly, one skilled in the art would not interpret either Blomgren or Blomgren2 as teaching instruction set engines for processing instructions from ISAs having different word sizes.

Applicants submit that the entire description of both Blomgren and Blomgren2 is devoid of any reference to providing processing for different ISAs having different ISA word sizes and more particularly, 32 bit and 64 bit word sizes. However, the case law is quite clear in establishing that each and every element of the claim must be exactly disclosed in the anticipatory

reference. Banner Titanium, *Id.* Hence, a *prima facie* case of anticipation of the claims over Blomgren and Blomgren2 has not been established and the rejection of Claims 1, 2, 4, 5, 7, 8 and 18 is therefore erroneous.

2. Specific Limitations Not Described in the Prior Art

Claims 9 and 17 recite a 32 bit size ISA mode and a 64 bit word ISA mode, which as noted by the Examiner is not taught or suggested by Blomgren.

3. Explanation Why Such Limitations Render the Claims Unanticipated by the Prior Art

Here, Blomgren provides no teachings with respect to a multi-mode processor that operates according to a first mode to process instructions from a 32 bit word ISA, referred to herein as a “legacy ISA”, as well as a 64 bit word ISA, referred to herein as a “current ISA”.

Applicants respectfully submit that the entire text of both Blomgren and Blomgren2 are devoid of any teachings with regards to processing instructions from ISAs having different 32-bit and 64-bit word sizes. Furthermore, as described above with reference to Claims 1 and 18, the Examiner improperly equates the term “instruction size” with the term “word size”.

However, the case law is quite clear in establishing that each and every element of the claim must be exactly disclosed by the combination of references cited by the Examiner. *Id.* Therefore, Applicants respectfully submit that a *prima facie* case of anticipation of Claims 9 and 17 is not established and therefore the rejection of Claims 9 and 17 is erroneous and should be overturned. Accordingly, Applicants respectfully request that the §102(b) rejection of Claims 9 and 17 be overturned.

E. Rejection of Claims 10-13, 15, 16 and 19 As Anticipated by Blomgren

The Examiner rejected all pending claims, including Claim 10-13, 15, 16 and 19 under 35 U.S.C. §102(e) as being anticipated by Blomgren.

1. Errors of Law and Fact in the Rejection

The Examiner has made the same errors as previously described with respect Claim 3. In addition, the Examiner has failed to show that each and every element of the independent Claims 10 and 19 are exactly disclosed by Blomgren. Banner Titanium. *Id.*

Applicants respectfully submit that the Examiner has incorrectly associated the term “unknown opcode” with the term “token” as recited by independent Claims 10 and 19. (*See* cols. 6-7, lines 61–10.)

According to the Examiner, remote control logic 42, which causes emulation mode to be entered whenever an unknown opcode is detected by instruction decode unit 36, as taught by Blomgren2, teaches the pre-processing hardware, as recited by Claim 3. (*See*, Blomgren2, col. 7,

lines 30-43.) Applicants respectfully submit that the detection of an unknown opcode, as taught by Blomgren2, neither teaches nor suggests detecting whether a token is received as an input of a plurality of floating point registers, as recited by Claims 10 and 19.

Furthermore, the passages cited by the Examiner to show pre-processing hardware within Blomgren refer to the sharing of data between registers, (Col. 2, lines 51-58 of Blomgren) and the use of two decode units for two ISAs that provide separate encodings (*See, Blomgren*, col. 1, lines 43-47). However, after carefully having reviewed these passages, Applicants respectfully submit that these passages clearly do not teach detecting whether a token is received as an input of a plurality of floating point registers, as recited by Claims 10 and 19.

Furthermore, Applicants submit that the cited passage provides no reference to whether a token is received as an input to determine what processing to take according to that input, depending on whether the processor is in a first mode or a second mode, as recited by independent Claims 10 and 19. Hence, a *prima facie* case of anticipation of the claims has not been established and the rejection of Claims 10-13, 15, 16 and 19 is therefore erroneous.

2. Specific Limitations Not Described in the Prior Art

Each of independent Claims 10 and 19 require:

fetching an input from at least one of a plurality of floating-point registers;
detecting whether the input includes a token;
if the token is detected in the input, checking what mode the processor is in;
if the processor is in a first mode, processing the input to render an
arithmetic result operation;
if the processor is in a second mode, performing a token specific operation.

(Emphasis added.)

3. Explanation Why Such Limitations Render the Claims Unanticipated by the Prior Art

The Applicants claim a method for processing an input to render an arithmetic result if a processor is in a first mode and performing a token specific operation if a processor is in a second mode when a token is detected as an input, as recited by Claims 10 and 19.

In contrast, Blomgren teaches the issuance of an unsupported opcode exception in response to detection of an unsupported CISC instruction and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation routine including various RISC instructions to perform the CISC instruction. Once the exception handling is complete, the mode is switched to the CISC mode and a next CISC instruction is executed. (*See*, col. 14, lines 22-28.)

Likewise, Blomgren2 teaches the detection of an unknown opcode and switches to an emulation mode when such an instruction is detected. The switch to the emulation mode results in the loading of an emulation driver to perform the CISC instruction. Once completed, the

emulation code returns control to the CISC mode, which causes a user program to continue execution with a next instruction. (See, col. 5, lines 57-65.)

In other words, both Blomgren and Blomgren2 teach the detection of an unknown opcode and switch to an emulation mode when such an opcode is detected. However, the passages cited by the Examiner, as well as the entire text of Blomgren and Blomgren2 are devoid of any reference to detecting whether an input from a plurality of floating point registers includes a token, as recited by Claim 3.

Furthermore, Blomgren and Blomgren2 are devoid of any teachings for detecting whether a token is received as an input to determine what processing to take according to that input, depending on whether the processor is in a first mode or a second mode, as recited by independent Claims 10 and 19. In contrast, Blomgren and Blomgren2 teach mode control logic that detects an unknown opcode which causes entry into emulation mode regardless of the processor mode, which the Examiner equates to a token, to load pointers with the address of an emulation routine. (See, Blomgren2, col. 7, lines 30-43.)

Accordingly, although Blomgren and Blomgren2 describe an emulation mode to emulate the unknown instruction, the processing performed within Blomgren and Blomgren2 does not vary according to a processor mode. Applicants respectfully submit that the detection of an unknown opcode and entry into emulation to emulate an unknown instruction does not teach the detection of whether an input contains a token or a token-specific operation according to a processor mode, as recited by Claims 10 and 19.

Therefore, a *prima facie* case of anticipation of the claims is not established and the rejection of claims should be overturned. *Id.* Accordingly, Applicants respectfully request that the §102(b) rejection of Claims 10-13, 15, 16 and 19 be overturned.

F. Rejection of Claim 6 and 14 As Anticipated by Blomgren and Blomgren2 in View of FOLDOC

1. Errors of Law and Fact in the Rejection

For the reasons provided below, the Examiner has failed to show that the prior art references of Blomgren and Blomgren2 in view of FOLDOC teach or suggest all claim features of Claims 6 and 14. The Federal Circuit Court of Appeals in In re Rijckaert, 9 F.3d 1531, 28 U.S.P.Q. 2d 1955 (Fed. Cir. 1993) held that:

In rejecting claims under 35 U.S.C. § 103, the examiner bears the initial burden of presenting a *prima facie* case of obviousness. . . . "A *prima facie* case of obviousness is established when the teaching from the prior art itself would appear to have suggested the claimed subject matter to a person of ordinary skill in the art." . . . If the examiner fails to establish a *prima facie* case, the rejection is improper and will be overturned. (Emphasis added.) 9 F.3d at 1532, 28 U.S.P.Q. 2d at 1956.

Applicants respectfully submit that the combined teachings of Blomgren and Blomgren2 in view of FOLDOC would not have suggested the claimed invention to one of ordinary skill in the art, as required to establish a *prima facie* case of obviousness. *Id.* Hence, a *prima facie* case of obviousness has not been established and the rejection is erroneous and should be overturned. *Id.*

According to the Examiner, Blomgren and Blomgren2:

have taught wherein the token represents a “not of the U thing” (NaTVal) that defines an unsuccessful load request (‘750 col. 4, lines 13-33 and col. 7, lines 25-32). (See, pg. 10, final paragraph of Office Action mailed 12/14/04.)

Applicants respectfully disagree with the Examiner’s contention. As described by Blomgren2:

Mode control logic 42 causes emulation mode to be entered whenever a miss is signaled from TLB 52, or where an unknown opcode is detected by instruction B code unit 36. (col. 7, lines 30-32.) (Emphasis added.)

Based on the cited passage above, the detection of a TLB miss does not involve some input operand, but instead is detected by remote control logic when a miss is signaled from TLB 52. Accordingly, although a miss within TLB 52, as taught by Blomgren2, is based on an unsuccessful load of a translation that is not present in TLB 52 (See, Blomgren2, col. 7, lines 25-29), the miss within the TLB does not involve some input operand that is set to NaTVal to notify remote control logic that an unsuccessful speculative load request is detected, as recited by Claims 6 and 14.

The Examiner further cites FOLDOC, which provides definitions of the terms “speculative evaluation” and “speculative execution.” According to the Examiner:

A person of ordinary skill in the art at the time the invention was made would have recognized that speculative evaluation and execution reduces the overall run time of a processor and keeps all functional units working, i.e., not wasted cycles, (FOLDOC terms “speculative evaluation” and “speculative execution”), thereby increasing processor speed and efficiency. Therefore, it would have been obvious to one of obvious skill in the art at the time the invention was made to incorporate the speculative evaluation and execution of FOLDOC in the device of ‘009 and ‘750 to include processor speed and efficiency. (See, pg. 11, first paragraph of Office Action mailed 12/14/04.)

Assuming that one of skill in the art would recognize that speculative evaluation and execution, as defined by FOLDOC, improved processor speed and efficiency, the Examiner fails to illustrate some rationale for combining the missing elements provided by FOLDOC within the teachings of Blomgren and Blomgren2.

It is well established that obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention absent the teaching or suggestion supporting such combination. ACS Hospital Sys., Inc. v. Montefiore Hospital, 732 F.2d. 1572,

1577, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984). Also, one cannot find obviousness through hindsight to construct a claimed invention from elements of the prior art. In re Warner, 379 F.2d 1011, 1016, 154 U.S.P.Q. 173, 177 (C.C.P.A. 1967).

Accordingly, Applicants' claimed invention could only be arrived at through inappropriate hindsight. Id. Accordingly, Applicants respectfully submit that the combined teachings of Blomgren, Blomgren2 and FOLDOC would not have suggested the claimed subject matter to one of ordinary skill in the art, as required to establish a *prima facie* case of obviousness. In re Rijckaert, *supra*. Hence, a *prima facie* case of obviousness has not been established and the rejection is erroneous and should be overturned.

2. Specific Limitations Not Described in the Prior Art

Claims 6 and 14 recite analogous claim features. Claim 6 is representative. Claim 6 recites:

wherein the token represents a "not a thing value" (NaTVal) that defines an unsuccessful speculative load request.

3. Explanation Why Such Limitations Render the Claims Non-obvious over the Prior Art

The Examiner fails to illustrate that the combination or modification of Blomgren and Blomgren2 in view of FOLDOC teaches or suggests each of the recited features of the claimed invention. However, the case law is clear in establishing that "to establish *prima facie* obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art." In re Royka, 490 F.2d 981, 180 U.S.P.Q. 580 (CCPA 1974).

Here, the claimed invention recites:

wherein the token represents a "not a thing value" (NaTVal) that defines an unsuccessful speculative load request. (Emphasis added.)

As indicated above, the issuance of a TLB miss, as taught by Blomgren2, does not involve a token, which would define a NaTVal as an unsuccessful speculative load request. Specifically, as taught by Blomgren2:

mode control logic 42 causes emulation mode to be entered whenever a miss is signaled from TLB 52. (Col. 7, lines 30-31.) (Emphasis added.)

Based on the cited passage above, mode control logic 42 does not process some input to detect whether a TLB miss has occurred, but instead receives a signal from TLB 52 when a TLB miss has occurred. Accordingly, although a TLB miss could refer to an unsuccessful load request, the absence of any input operand to communicate such information to mode control logic fails to provide a teaching or suggestion of a token that represents an NaTVal value defining an unsuccessful speculative load request, as recited by Claims 6 and 14.

Accordingly, Applicants respectfully submit that even assuming, *arguendo*, that FOLDOC, as cited by the Examiner, teaches speculative execution and evaluation, the absence of some token associated with a TLB miss and modification of such a token to represent an NaTVa1 that defines an unsuccessful speculative load request is not possible. Accordingly, Applicants respectfully submit that the Examiner fails to establish a *prima facie* case of obviousness of the claimed invention, since the Examiner fails to illustrate the teachings or suggestion of all claim limitations of Claims 6 and 14 within the prior art. *Id.*

Furthermore, Applicants respectfully submit that the Examiner is engaged in a hindsight-based analysis to provide some rationale for combining the missing elements of FOLDOC with the teachings of Blomgren and Blomgren2. As a result, Applicants respectfully submit that the Examiner fails to establish that it would be obvious to combine the missing elements provided by FOLDOC with the teachings of Blomgren and Blomgren2. *In re Warner, supra.*

Accordingly, Applicants respectfully submit that the combined teachings of Blomgren, Blomgren2 and FOLDOC would not have suggested the claimed invention to one of ordinary skill in the art as is required to establish a *prima facie* case of obviousness. *In re Rijckaert, supra.* Hence, a *prima facie* case of obviousness has not been established and the rejection is erroneous and should be overturned. *Id.* Accordingly, Applicants respectfully request that the §103(a) rejection of Claims 6 and 14 be overturned.

G. Rejection of Claim 18 As Obvious Over Blomgren and Blomgren2 in View of Omondi

The Examiner rejected pending Claim 18 as obvious over Blomgren and Blomgren2 in view of Omondi.

1. Errors of Law and Fact in the Rejection

The Examiner has made the same errors as previously described with reference to rejected Claims 1, 2, 4, 7, 8 and 18, which the Examiner rejected under 35 U.S.C. §102(b) as anticipated by Blomgren and Blomgren2.

Applicants respectfully submit that the combined teachings of Blomgren and Blomgren2 in view of Omondi would not have suggested the claimed invention to one of ordinary skill in the art, as required to establish a *prima facie* case of obviousness. *Id.* Hence, a *prima facie* case of obviousness has not been established and the rejection is erroneous and should be overturned. *Id.*

Regarding the Examiner's citing of Omondi to show a plurality of floating point registers, Applicants respectfully submit that the Examiner's citing of Omondi fails to rectify the deficiencies attributed to Blomgren and Blomgren2 to teach a multimode processor, which includes

a plurality of instruction set engines to process instructions from a plurality of instruction set architectures having different word sizes, as recited by Claim 18.

Hence, the failure of Omondi to teach a plurality of ISAs having different word sizes, illustrates that the combined teachings of Blomgren and Blomgren2 in view of Omondi would not have suggested the claimed subject matter to one of ordinary skill in the art, as required to establish a *prima facie* case of obviousness. Id. Hence, a *prima facie* case of obviousness has not been established and the rejection is erroneous and should be overturned. Id.

2. Specific Limitations Not Described in the Prior Art

Claim 18 recites:

a plurality of instruction set engines to process instructions from a plurality of instruction set architectures having different word sizes;
a mode identifier;
a plurality of floating-point registers shared by the instruction set engines;
and
a plurality of floating-point units coupled to the floating-point registers, the floating-point units processing an input responsive to the mode identifier.

3. Explanation Why Such Limitations Render the Claims Non-obvious over the Prior Art

The Examiner fails to illustrate that the combination or modification of Blomgren and Blomgren2 in view of Omondi teaches or suggests each of the recited features of the claimed invention. However, the case law establishes that a *prima facie* case of obviousness requires all claim limitations to be taught or suggested by the prior art. In re Royka, supra. Here, the claimed invention recites:

a plurality of instruction set engines to process instructions from a plurality of instruction set architectures having different word sizes.

As indicated above, Omondi provides no teachings or suggestions with regards to ISAs with different word sizes. Furthermore, as indicated above, the Examiner incorrectly equates the term “instruction size” with the term “word size.” As defined by Exhibit 1, word size refers to the number of bits that a CPU can process at one time. (*See*, Exhibit 1.) As indicated above, the number of bits a CPU can process at one time does not vary depending on whether the CPU is processing CISC instructions or RISC instructions.

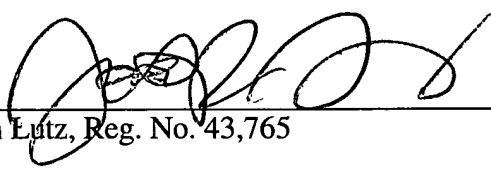
Accordingly, Applicants respectfully submit that the combined teachings of Blomgren, Blomgren2 and Omondi would not have suggested the claimed invention to one of ordinary skill in the art, as required to establish a *prima facie* case of obviousness. In re Rijckaert, supra. Hence, a *prima facie* case of obviousness has not been established and the rejection is erroneous and should be overturned. Id.

Based on the foregoing, Applicant requests that the Board overturn the rejection of all pending claims and hold that all of the claims of the present application are allowable.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN

Dated: July 11, 2005



Joseph Lutz, Reg. No. 43,765

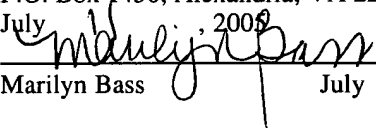
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(310) 207-3800

CERTIFICATE OF MAILING:

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail, with sufficient postage, in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on

July

2005



Marilyn Bass

July 11, 2005

Attachment: Exhibit 1

VIII. CLAIMS APPENDIX

The claims involved in this Appeal are as follows:

1. (Previously Presented) A processor comprising:
 - a first instruction set engine to process instructions from a first instruction set architecture (ISA) having a first word size;
 - a second instruction set engine to process instructions from a second ISA having a second word size, the second word size being different than the first word size;
 - a mode identifier;
 - a plurality of floating-point registers shared by the first instruction set engine and the second instruction set engine; and
 - a floating-point unit coupled to the floating-point registers, the floating-point unit processing an input responsive to the mode identifier to produce an output.
2. (Original) The processor of Claim 1 wherein the mode identifier is one of a plurality of bits in a processor status register.
3. (Previously Presented) The processor of Claim 1 wherein the floating-point unit comprises:
 - pre-processing hardware to detect if a token exists in the input;
 - an arithmetic unit responsive to the input and the mode identifier; and
 - post-processing hardware to perform a token specific operation if a token exists in the input.
4. (Previously Presented) The processor of Claim 1 wherein the input includes data stored in at least one of the floating-point registers.
5. (Previously Presented) The processor of Claim 1 wherein the input may contain a token, wherein the floating-point registers are 82 bits wide, and wherein the token being an 82 bit processor known value.
6. (Previously Presented) The processor of Claim 3 wherein the token represents a “not a thing value” (NaTVal) that defines an unsuccessful speculative load request.

7. (Original) The processor of Claim 1 wherein the floating point registers each comprise:

- a sign bit,
- an exponent; and
- a significand.

8. (Original) The processor of Claim 1 wherein the mode identifier indicates whether the processor is in a first mode or a second mode.

9. (Previously Presented) The processor of Claim 1 wherein the mode identifier indicates whether the processor is in a 32 bit word ISA mode or a 64 bit word ISA mode.

10. (Previously Presented) A method in a processor comprising:
fetching an input from at least one of a plurality of floating-point registers;
detecting whether the input includes a token;
if the token is detected in the input, checking what mode the processor is in;
if the processor is in a first mode, processing the input to render an arithmetic result;
if the processor is in a second mode, performing a token specific operation; and
producing an output.

11. (Previously Presented) The method of Claim 10 wherein the input is comprised of at least one operand and at least one operator; wherein detecting comprises examining the at least one operand to determine whether any of the operands correspond to the token; and wherein checking comprises examining a mode identifier to determine whether the processor is in the first mode or the second mode.

12. (Previously Presented) The method of Claim 10 wherein processing comprises executing at least one operation on the at least one operand according to the at least one operator to achieve a result.

13. (Original) The method of Claim 10 wherein performing comprises propagating the token; and wherein producing output comprises setting the output to be the token.

14. (Original) The method of Claim 10 wherein the token represents a “not a thing value” (NaTVal) that defines an unsuccessful speculative load request.

15. (Original) The method of Claim 10 wherein checking comprises checking a mode identifier.

16. (Original) The method of Claim 10 wherein checking comprises checking a mode identifier bit in a processor status register.

17. (Original) The method of Claim 11 wherein the first mode is a 32 bit word ISA mode and the second mode is a 64 bit word ISA mode.

18. (Previously Presented) A multi-mode processor comprising:
a plurality of instruction set engines to process instructions from a plurality of instruction set architectures having different word sizes;
a mode identifier;
a plurality of floating-point registers shared by the instruction set engines; and
a plurality of floating-point units coupled to the floating-point registers, the floating-point units processing an input responsive to the mode identifier.

19. (Previously Presented) A method in a multi-mode processor comprising:
fetching an input from at least one of a plurality of floating-point registers;
detecting whether the input includes at least one token of a plurality of tokens;
if at least one token is detected in the input, checking what mode the processor is in;
processing the input to render an arithmetic result when the processor is in at least a first mode of a plurality of modes; and
performing a token specific operation when the processor is in at least a second mode of a plurality of modes.

Computer Dictionary Online

[Medical Dictionary](#) [Law Dictionary](#) [Legal Dictionary](#)

[0](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [a](#) [b](#) [c](#) [d](#) [e](#) [f](#) [g](#) [h](#) [i](#) [j](#) [k](#) [l](#) [m](#) [n](#) [o](#) [p](#) [q](#) [r](#) [s](#) [t](#) [u](#) [v](#) [w](#) [x](#) [y](#) [z](#)

Meet Jewish Singles

Free to Join. 1000's of pictures
& video's of Beautiful Jewish
singles

New DOT 5th Edition

New Dictionary Of Occupational
Titles 5th Ed w/O*NET Def
04pp \$149

Talking Dictionary

Great deals on Talking
Dictionary Shop on eBay and
Save!

dictionaryare.com

Teach yourself Romanian
language with this software
package

[Goooooogle-Anzeigen](#)

word size

<processor> The number of bits that a CPU can process at one time. Processors with many different word sizes have existed though powers of two (8, 16, 32, 64) have predominated for many years. A processor's word size is often equal to the width of its external data bus though sometimes the bus is made narrower than the CPU (often half as many bits) to economise on packaging and circuit board costs.

(1995-04-23)

[Contact the Computer Dictionary Online](#) :: [Link to the Computer Dictionary Online](#) :: [Disclaimer for Computer Dictionary Online](#)

Computer Dictionary Online
Copyright © 2005